논·문 17-1-2

무선 환경에서 데이터 공간의 확장을 위한 모바일 에이전트의 설계 및 구현 (Design and Implementation of Mobile Agent for Extension of Data Space in Wireless Environment)

나 승원, 오 세만
동국대학교 컴퓨터공학과
{nasw, smoh}@dgu.ac.kr

요 약

이동통신과 인터넷 기술의 결합으로 탄생한 무선 인터넷은 이동의 편리성을 제공하며 성장해 가고 있다. 그러나 모바일 환경이 가지고 있는 제한 사항 때문에 대중적인 서비스로 발전하지 못하고 있는 실정이다. 특히 모바일 환경의 제한 요소 중 협소한 메모리 공간으로 효율적인 자원 관리를 수행하지 못하는 단점이 있다. 기술이 발전함에 따라서 메모리 공간을 확대시켜 이와 같은 문제를 해결할 수는 있겠지만 소형화되어야 하는 디바이스의 하드웨어 특성상 충분한 메모리 용량을 확보하는 데에는 한계가 있다.

본 논문에서는 모바일 디바이스의 내부에서 외부의 서버까지 메모리 공간을 확장하여 데이터 활용이 가능하고 디바이스 내부의 파일을 효율적으로 관리할 수 있는 모바일 에이전트를 제시하며 이를 실현하기 위한 자원 관리시스템(RMS: Resources Management System)을 설계하고 구현하였다. 제안된 RMS를 적용한 디바이스는 데이터 영역의 확장을 통해서 확대된 프로세스 적용이 가능하며 내부 파일을 관리하여 최적의 메모리 공간을 유지하는 효과를 제공한다.

1. 서론

인터넷이 결합된 이동통신 기술은 무선인터넷 서비스를 창출하였고 이동의 편리성이라는 장점을 가지고 대중적인 인터넷 서비스로 성장해 가고 있는 추세이다. 국내의 무선 인터넷 서비스 가입자가 2002년 12월말 현재 2천9백만 명으로 전체 휴대전화 가입자의 89%를 점유하고 있으나 월 평균 60분 이상을 이용하는 사용자는 무선 가입자의 10%를 넘지 못한다[11]. 이와 같이 모바일 환경은 작은 디바이스의 처리 능력과 통신 속도의 문제를 가지고 있어서 무선 인터넷이 대중화되는 데에는 많은 시간과 노력이 요구된다.

특히 협소한 메모리 공간 때문에 고객이 원하는 컨텐츠 파일을 저장하여 실행하기에는 제한되어 있어 디바이스의 용량에 적합한 크기로 축소하고 있는 실정이다. 따라서 디바이스 내부에서는 파일의 자원 관리를 효율적으로 수행하지 못하여 최적의 서비스 구현에는 한계가 있다[3]. 고객은 이미 PC에서 인터넷을 경험하였고 그 기능들을 무선인터넷에서도 요구하고 있는 실정이다. 그러나 모바일 디바이스가 최소화되는 경향이므로 기술 개발이 이루어지더라도 충분한 메모리 공간을 확보한다는 것은 하드웨어 특성상 어려운 사항이다. 따라서 무선 인터넷을 수행하기 위한 모바일 디바이스는 필요한 메모리 영역을 디바이스 내부 뿐 아니라 외부의 서버 영역까지 확대하여 프로세스를 수행하는 구조로 발전되어야 할 것이다.

본 논문에서는 인터넷 수행을 위한 모바일 디바이스의 메모리 공간을 외부의 영역까지 확장하여 데이터 활용을 가능하게 하고 디바이스 내부의 파일을 효율적으로 관리하기 위한 모바일 에이전트를 제안하며 이를 실현하기 위한 자원 관리시스템(RMS: Resource Management System)을 설계하고 구현하였다. 제안된 RMS를 적용한 디바이스는 외부로의 데이터 영역의 확장으로 “모바일 공간 확장”의 개념을 제안하여 확대된 프로세스 적용이 가능하며 내부 파일을 관리하여 최적의 메모리 공간을 유지하는 효과를 가진다.

2. 관련연구

2.1 분산 에이전트

에이전트 기술은 사용자가 수행 할 작업이나 반복적인 일 들을 자동적으로 처리해 주는 시스템이다[7]. 에이전트는 지시된 작업을 수행하고 조정하기 위한 제어지식 에이전트와 수행할 역할의 기능을 규정하고 있는 영역지식 에이전트, 그리고 프로시저 콜과 메시지 교환을 처리하는 통신모듈 에이전트로 구성된다. 분산 컴퓨팅 환경에서의 에이전트는 인터페이스, 태스크(Task), 정보 에이전트로 구분한다[2]. 인터페이스 에이전트는 사용자와 직접적으로 상호 작용을 하면서 사용자의 요구 사항을 분석한다. 태스크 에이전트는 영역 지식을 기반으로 사용자의 요구사항을 직접 수행한다. 그리고 정보 에이전트는 여러 곳에 흩어져 있는 정보에 접근할 수 있는 기능을 제공하는 에이전트이다.

무선 환경에서의 에이전트 구성은 사용자가 필요로 하는 정보를 검색하여 보여주는 형태로서 다운로드 시스템이 이 경우에 해당된다. 무선 에이전트는 오퍼레이션시스템 위에서 해당 애플리케이션과 인터페이스 관계를 형성하며 수행을 유도한다.

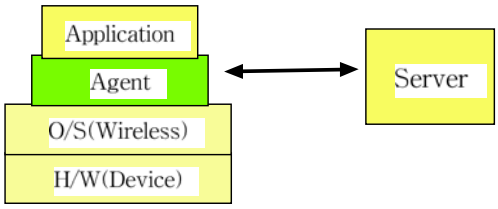

그림 1. 무선 환경에서의 에이젼트

2.2. 분산 객체 기술

분산 객체(Distributed Object)는 다른 메모리에 존재하고 있는 객체를 네트워크를 이용하여 호출하는 형태로서 분산 객체는 일반

객체와 마찬가지로 데이터베이스와 메소드로 이루어져 있으며 플랫폼과 프로그램 언어에 대한 독립성을 제공한다[10]. 분산 객체 기술은 객체 지향과 객체간 메시지 교환 기술의 장점을 효과적으로 통합하여서 개발자에게는 어플리케이션 개발의 생산성을 제공하고 사용자에게는 분산 환경의 투명하고 통합된 정보를 제공한다. 분산 객체의 배경 기술로는 객체지향 기술, 미들웨어 기술 등이 있으며 대표적인 분산 객체로는 RMI와 DCOM등이 있다[10].

2.3.1 RMI(Remote Method Invocation)

RMI는 자바 기반의 분산 객체 기술로서 분산되어 있는 객체를 마치 로컬에 있는 객체로 사용할 수 있으며 기존의 소켓 프로그램과 같은 복잡한 설계 절차와 에러를 발생시키는 요인을 극소화시켰다. 소켓 API를 사용해서 구현되는 부분은 RMI 패키지, 클라이언트 스터브(Client Stub) 그리고 서버 스터브(Server Stub)에 의해 포함된다. RMI는 RPC(Remote Procedure Call)와는 달리 JVM 환경에서 실행되므로 자바 이외의 다른 언어는 지원하지 않는다. RMI의 주요 기능은 원격 호출을 제공하고, 서버에서 애플릿의 콜백(Callback) URL 적용이 가능하다. 또한 분산 모델을 자바 환경으로 통합할 수 있으며 분산 객체와 비 분산 객체의 명확한 구분이 가능한 기능을 제공한다.

2.3.2 DCOM(Distributed COM)

DCOM은 원격 객체 액세스를 지원하는 마이크로 소프트의 기술이다. 윈도우 환경에서 컴포넌트간의 연동을 위한 분산 객체 시스템으로 네트워크에서 원격 객체가 제공하는 서비스들을 요청하기 위해서 클라이언트 응용 프로그램을 사용하도록 지원한다. COM(Component Object Model)은 같은 기계에서 컴포넌트간의 상호 작용만을 지원하지만 DCOM은 이러한 COM의 제약을 해결하여 분산 컴퓨팅 환경에서도 컴포넌트간의 상호 작용을 가능하게 하였다. COM에서는 제공하지 않는 위치 투명성과 원격 활성화, 객체들간의 연결관리, 그리고 보안 기능이 DCOM에서 추가된 사항이다[7].

그림 2에서 DCOM의 컴포넌트간 연결 프로토콜은 RPC(Remote Procedure Call)를 기반으로한다. DCOM 프로토콜의 상위 계층인 ORPC(Object RPC)는 객체 지향 RPC를 지원하는 프로토콜을 의미한다.

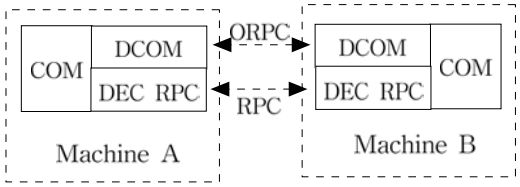

그림 2. RPC 기반의 DCOM 구조

3. RMS의 설계

3.1 RMS의 개요

RMS는 데이터 저장 공간을 서버의 영역까지 확대하여 사용할 수 있는 기술과 디바이스 내부의 메모리 자원 관리를 효율적으로 수행하기 위한 모바일 에이전트이다. RMS를 통해서 “모바일 공간 확장”의 개념을 제안하고자하며 구체적인 설계 방법으로 분산

객체와 3-Tier 기술을 적용하였다.

RMS의 기본 구조는 RMS 에이전트, RMS 도메인, RMS 서버로 구성되며 그림 3과 같은 시스템 구성도를 가진다. RMS를 통하여 호출되는 리소스 파일들은 DLL (Dynamic Linking Library) 파일과 OCX (OLE Control Extension) 파일, 그리고 컨텐츠 파일을 대상으로 한다

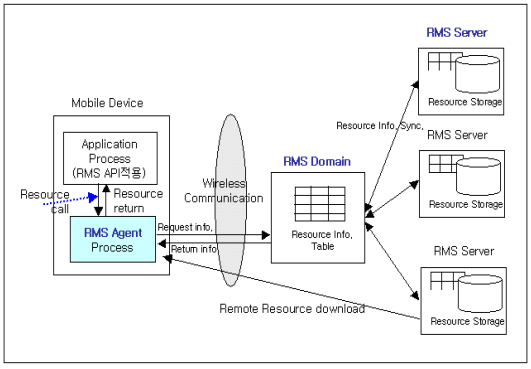

그림 3. RMS 시스템 구성도

RMS 에이전트는 디바이스 내부에서 수행되며 실행 파일과 링크 될 리소스 파일을 RMS 도메인으로 호출한다. RMS 도메인은 등록된 리소스 파일을 포함한 오브젝트의 원격 주소를 관리하고 해당 오브젝트가 존재하는 서버로 파라미터를 전달하여 에이전트와 서버간의 원활한 로드 밸런싱 기능을 제공한다. RMS 서버는 해당 리소스 파일인 오브젝트가 존재하는 서버로서 에이전트에게 해당 리소스 파일을 제공한다. RMS는 리소스 파일의 위치 투명성과 원격 객체의 동적 활성화, 객체의 수명 관리 및 동적 인터페이스 검색, 그리고 바이너리 호환성등의 기능을 제공하는 것에 목적을 둔다. 따라서 이러한 기능을 제공하기 위해서 RMS 에이전트에서는 실행 파일의 호출 정보를 그룹 별로 분류하기 위한 RMS 전용 API Set을 정의하고 해당 리소스를 호출하는 프로세스 확장 모듈을 제공한다.

그림 4는 RMS를 수행하는 일련의 과정들을 다이어그램으로 표현한 사항이다. 즉 필요한 리소스 파일을 호출하여 수행하고 내부 공간을 조절하는 형태의 다이어그램이다.

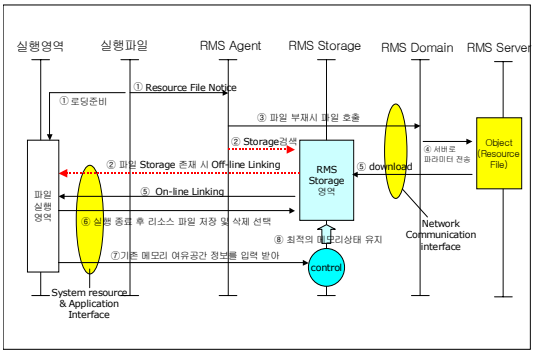

그림 4. RMS Diagram

3.2 RMS 에이전트의 설계

RMS 에이전트는 디바이스 내부에 존재하는 애플리케이션 프로세스의 리소스 호출 범위를 로컬에서 리모트로 확장하기 위한 프로세스이다. 이것은 데몬(Demon) 형태의 프로세스로 메모리에 상주하며 디바이스와 서버간의 중계 역할과 로컬의 파일을 관리하는 기능을 수행한다.

아래 그림 5는 모바일 디바이스의 실행 환경에서 OS 프로세스 및 애플리케이션 프로세스에 필요한 리소스를 호출하게 되며 이에 대한 리소스의 반환은 해당 프로세스가 로컬 리소스 스토리지 범위 내에서만 검색하여 반환하는 구조로 수행되어진다. 즉, 기존의 모바일 디바이스 실행 형태를 나타내고 있는

사항이다.

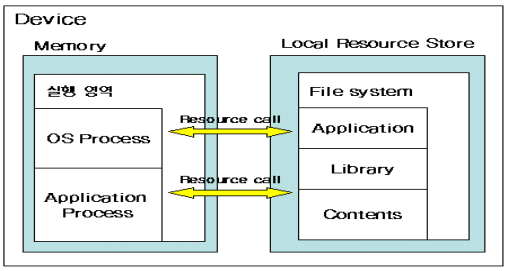

그림 5. 기존 프로세스 수행구조

그림 6은 제안하는 RMS 에이젼트의 수행 모델로서 RMS 에이전트 프로세스와 RMS API set으로 구성되고 RMS 에이전트 프로세스는 RMS API가 적용된 경우이며 애플리케이션 프로세스가 요청하는 로컬 및 리모트 상의 리소스를 관리하여 반환하는 구조로 설계하였다.

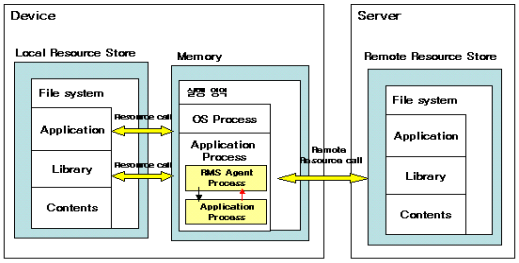

그림 6. RMS의 리모트 프로세스 수행구조

3.2.1 RMS 에이전트 구성

RMS 에이전트는 애플리케이션 프로세스의 리소스 요청에 대한 처리를 수행하는 서브 에이전트 프로세스 부분과 로컬에 저장된 리소스를 관리하는 리소스 스토리지 프로세스 부분으로 구성된다.

첫째, 서브 에이전트 프로세스는 디바이스에서 수행되는 실행 모듈이 메모리 환경에서 애플리케이션 프로세스 형태로 수행되는 것이며 리소스 파일을 호출 할 경우 RMS 에이전트 프로세스가 리소스 호출 부분의 수행을 대신하여 리소스의 위치 정보를 검색한 후에 해당 리소스 핸들을 애플리케이션 프로세스로 반환하는 역할을 수행한다. 리소스 파일의 저장은 RMS가 관리하는 별도의 스토리지에 다운로드 된다. 리소스 파일을 검색하는 기준은 RMS 도메인과 RMS 서버의 리소스 로케이션 매핑 테이블을 참조한다. 리소스 로케이션 매핑 테이블은 도메인 설계 시 표현하였다. 그림 7과 같이 4가지 기능으로 에이전트 프로세스의 역할을 수행한다.

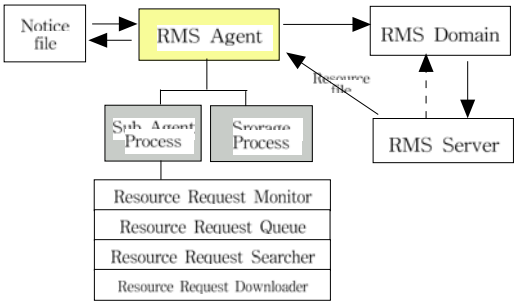

그림 7. RMS Sub Agent Process Area

둘째, 리소스 스토리지 프로세스는 다운로드 된 리소스 파일을 관리하는 영역으로 실행 종료 후 리소스 파일을 RMS 스토리지에 저장하거나 삭제할 수 있으며 디바이스 내부의 메모리 공간 정보를 OS로부터 입력받아서 RMS 저장 공간을 조절하여 최적의 메모리 상태를 유지하여 효율적인 자원관리 기능을 제공한다. 스토리지 내부의 파일 관리 기준은 오래된 파일부터 삭제하고 동일 조건 시 버전 정보나 파일 크기의 순으로 관리하는 일반적인 방법론을 적용하였다.

RMS 스토리지 공간은 리소스 영역과 컨텐츠 영역으로 디렉토리를 구성하여 각각의 디렉토리에 해당 파일을 관리한다.

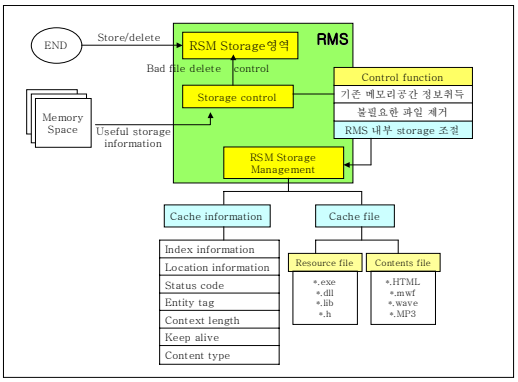

그림 8. Resource Storage Process Area

종합적으로 RMS 에이전트는 다운로드 수행기능의 프로세스와 리소스 스토리지 프로세스의 기능으로 구분하며 리모트 상의 리소스도 로컬 영역의 리소스처럼 처리하여 외부의 메모리 공간을 확장할 수 있으며 이것은 로컬 로케이션이나 리모트 로케이션에 상관없이 리소스를 호출할 수 있는 기본 설계의 개념이 되는 것이다. 이 기능을 원활히 수행하기 위해서는 범용성을 제공하는 RMS API Set을 구성하여 외부 영역의 확장을 구체화하였다.

3.2.2 RMS 에이전트 API set 구성

RMS 에이전트 API set은 기존 디바이스의 OS에 독립적인 native API의 특정 부분에 대해서 API를 보완하는 개념이다. RMS 에이전트 API Set은 다음과 같다.

추가된 API의 기능은 Native API에서 로컬 영역의 리소스 호출에 대한 리모트 호출 부분을 지원하는 기능으로 확장되어 있다. 다음 예제는 RMS API Set을 적용한 RMS Agent 소스 코드의 일부이다.

RMS API Set	Function
RMS_Create()	RMS 실행을 위한 메인 인터페이스 함수
RMS_LoadResource()	실행 파일이 수행하기 위한 리소스 파일 호출
RMS_Search_Location()	리소스의 위치를 검색하여 위치 정보 제공
RMS_Link_App()	Storage에 있는 리소스 파일이 기존의Application과 Link됨
RMS_Get_Resource()	RMS_Search_Localtion의 위치정보에 따른 리소스 handle 반환
RMS_Process_Type()	리소스의 Type 반환 및 관련 Application 정보 제공
RMS_VM_Interface()	리소스 호출 후 기존 내장되어 있는 VM과의 호완성 유지

[표 1] RMS 에이전트 API set

3.3 RMS 도메인의 설계

RMS 도메인에서는 RMS 에이전트로부터 호출받은 리소스의 위치 정보를 체계적으로 분류하고 해당 데이터가 있는 서버와 에이전트를 연결하는 인터페이스를 제공한다. 즉, 중간 디렉토리 서버로서의 2차 에이전트 기능을 수행하고 있다.

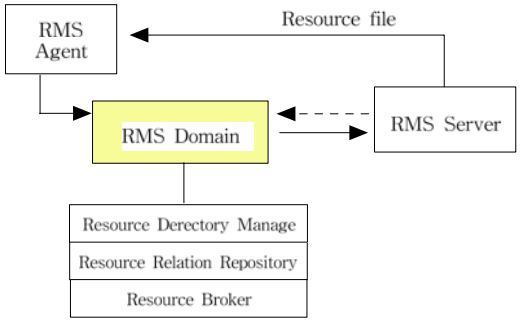

그림 9. RMS 도메인의 구조

RMS 도메인은 그림 9와 같이 3가지의 기능을 제공한다.

```
// test.exe 프로그램에서 test.dll을 remote call 하여
// test.exe 프로그램 프로세스에 Load하는 프로그램

#include "stdafx.h"
#include "RMS_Library.h"
RMS_Create()//RMS 실행을 위한 메인 인터페이스 함수
int RMS_LoadResource(szfileName,
                szfileName,
                szCmdLine,
                NULL,
                NULL, FALSE,
                dwCreationFlags,
                NULL,NULL,NULL, &pi) {
                INT rc;
                switch (RMS_Search_Location(szfileName)) {
case IS_Remote_TCP :  // Remote 위치일 경우
RMS_Get_Resource(szfileName);
     // 도메인 서버의 중재로 RMS Agent와 RMS 서버와의
     // Resource download 시행 후 Resource memory load
if (Process_Type(szfileName) == IS_PROCESS)
     // 메모리에 Load된 Resource 타입이 Program일 경우
     rc = CreateProcess (szfileName, szCmdLine,
          NULL, NULL, FALSE,
          dwCreationFlags, NULL,
          NULL,NULL, &pi); // 신규 out-process 생성
     if (Process_Type(szfileName) == IS_LIBRARY){
       //메모리에 Load된 Resource 타입이 Library일 경우
          HINSTANCE hInst;
          int (*pFunc)(LPCTSTR szName);
     if ((hInst = LoadLibrary(szParameter)) == NULL)
       // 신규 in-Process load.
     {
       LPVOID lpMsgBuf;
       Message(Message_FORMAT);
       LocalFree( lpMsgBuf );
       return 0; }
     }
  if (Process_Type(szfileName) == IS_CONTENTS)
     // 메모리에 Load된 Resource 타입이 Contents일 경우
    CreateProcess(L_MOUNT_APP, szParameter, NULL,
    NULL, NULL, 0, NULL, NULL, NULL, &pi);
     // contents file과 연계되는 외부 out-process 생성 후
     // 컨텐츠 파라미터 전달
  break;
case IS_LOCAL_CACHE : // Local Cache 위치일 경우
  break;
case IS_LOCAL_IN : //  Local 위치일 경우
  break;
case IS_NON : // 위치 정보 값이 없을 경우
  break;
               }
return rc;}
int WINAPI WinMain( HINSTANCE hInstance,
                    HINSTANCE hPrevInstance,
                    LPTSTR    lpCmdLine,
                    int   nCmdShow)
{
     TCHAR szfileName[MAX_PATH];
     TCHAR szCmdLine [64];
     DWORD dwCreationFlags;
     PROCESS_INFORMATION pi;
     INT rc;
     lstrcpy (szfileName, TEXT ("test.dll"));
  /* 메인 프로세스에서 test.dll 호출
     lstrcpy (szfileName, TEXT ("test.bmp")); //Image 호출
     lstrcpy (szfileName, TEXT ("test.mp3")); //Sound 호출
     lstrcpy (szfileName, TEXT ("test.wmv")); //movie 호출
  */
    lstrcpy (szCmdLine, TEXT (""));
     dwCreationFlags = 0;
rc = RMS_LoadResource (szfileName, szCmdLine,NULL,
          NULL, FALSE,
         dwCreationFlags, NULL,NULL,NULL, &pi);
           // 메인 프로세스에서 test.dll을 로드하기 위해
           // RMS Agent API의 Resource Load 함수를 호출
             if(rc) {
                  CloseHandle (pi.hThread);
                 CloseHandle (pi.hProcess);
  }
     return 0;
}
```

첫째, 리소스 디렉토리 매니징 기능은 등록된 객체 파일의 원격 서버 주소나 로컬 디렉토리, 파일이름, 리소스 파일의 주소 값을 관리하는 기능을 제공한다. 둘째, 리소스 릴레이션 레파지토리(Repository)의 기능은 리소스 파일의 GUID와 버전 정보, 부가 정보 및 관련 객체의 GUID를 제공하여 준다. 세째, 리소스 브로커의 기능은 RMS 에이전트와 RMS 서버간의 최적의 경로를 검색하여 해당 파일을 에이전트에게 연결하는 기능을 지원한다. 그리고 다수의 에이전트가 동시에 접속 할 경우 접속 순위를 결정하여 네트워크 경로 및 RMS 서버 부하에 대한 에러 복구에 대한 처리 기능과 확장성 지원 그리고 파일 전송에 대한 로드 밸런싱 기능을 제공한다.

[표 2] RMS 파일정보 매핑 테이블

구 분	ID	상 세 정 보
무선 접속인증	01	고객 ID, 전화번호(MIN)
디바이스 종류	02	PDA, Hand-phone, other
디바이스 O/S	03	PPC 2002, .Net. Palm, Linux
리소스 파일	04	▷ Resource : dll, ocx ▷ contents : App, 컨텐츠

리소스 디렉토리 매니징 기능 중 RMS 에이전트로부터 호출 받은 접속 데이터 값을 RMS 서버로 전송하기 위해서는 해당 리소스 파일 정보를 관리하는 고유 ID 관리 테이블이 존재해야하며 표 2는 대략적인 ID 구조를 나타낸 것이다.

다음 코드는 호출 받은 RMS.dll을 서버로 연동하기 위한 브로커 영역에서의 코드 형식이다.

```
 //Resource Broker부분 code
InitServerPath();
  //도메인에 등록되어 있는 서버 리스트 초기화
InitInfoSync();
  // 도메인에 등록되어 있는 서버와의 정보 동기화
while(FaultDomain()) //에러가 없는 동안 Loop 수행
{
   rqParam= Get_Message(rqMessage);
  //도메인에 요청된 리소스 call 메시지 수신
   if(ServerLoadBalance(rqParam))
  /* 수신된 메시지에서 요청한 file이 위치하는 서버
  검색 및 최적 경로 설정*/
   {
   Broke_AgetServer(rqParam);
   //에이전트와 서버간의 연결 설정
   }
   else {
      Message(Message_FORMAT);
   }
}
```

3.4 RMS 서버의 설계

RMS 서버는 RMS 도메인으로부터 리턴 받은 호출에 대해서 해당 리소스를 RMS 에이전트에 제공하며 실제로 파일을 가지고 있는 서버이다. RMS 서버를 구성하는 세 가지 주요 기능은 접속 모듈을 담당하는 커넥션 매니지먼트(Connection Management)와 파일 버전 정보와 체계적 관리를 위한 리소스 매니지먼트(Resource Management), 그리고 리소스 파일이 존재하는 스토리지 영역(Storage Area)이 있다.

첫째, 커넥션 매니지먼트의 기능은 RMS 도메인을 통해서 호출한 RMS 에이전트와의 패킷 전송으로 연결 관리를 수행한다.

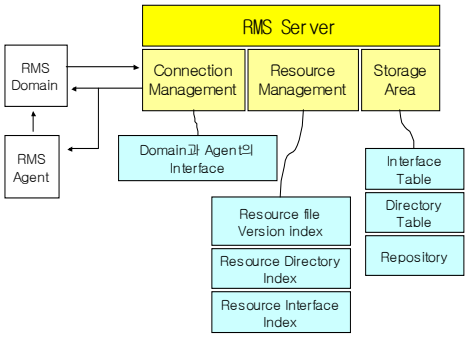

그림 9. RMS 서버의 구조

이것은 모든 RMS 에이전트에 개방되어 RMS 서버가 응답할 때 전체 네트워크의 성능 저하를 초래하기 때문에 RMS 도메인에서 연결된 RMS 에이전트와 전용으로 인터페이스된다. 그리고 RMS 도메인의 리소스 릴레이션 레파지토리와 리소스 버전 컨트롤을 설정하여 최신의 파일 정보를 실시간으로 제공해 준다. 둘째, 리소스 매니지먼트에서는 리소스 파일의 등록 및 버전 관리 기능과 디렉토리 인덱스 관리, 인터페이스 인덱스 관리 등의 기능이 수행되어 전용 모바일 디바이스 타입, 리소스 파일 타입, 리소스 파일의 버전 및 파일 크기 등을 관리한다. 셋째, 스토리지 영역은 제공 될 리소스 파일이 위치하는 곳으로 디렉토리 테이블과 인터페이스 테이블, 레파지토리 등을 관리하는 기능을 제공한다. 다음 코드는 RMS 서버에서 커넥션 매니지먼트에 관련된 코드의 일부 사항이다.

```
//Connection Management code 부분
while(FaultServer())
{
   Init_PathInfo(); // 도메인에 연결 설정 초기화
   if(Init_ConnServerAgent(rqParam))
   //에이전트와 서버간의 연결 초기화 및 파일 전송 준비
   {
     if(Init_VersionControl(rqParam)){
    // 요청한 파일과 전송할 파일의 버전 컨트롤 수행
       if (Sendfile(rqParam)){
    // 신규 버전 파일을 에이전트에 전송
    // 파일 전송 에러 처리 구문
       }
     }
    }
   else {
     Message(Message_Format));
   }
}
```

4. RMS의 구현

4.1 개발환경 및 구현도구

본 논문에서 제안한 RMS를 구현하기 위해서 PDA(Personal Digital Assistants)를 선정하였다. 이동의 편리성은 휴대폰이 유리하지만 디바이스의 확대된 기능을 제공하는 것은 PDA가 우월하다. 향후의 무선 인터넷 모바일 디바이스는 휴대폰의 크기와 PDA의 성능이 결합된 모델로 제공될 것이다.

RMS를 적용하기 위한 개발 환경 및 구현 도구는 표 3과 같다.

[표 3] 개발환경 및 구현도구

구 분		세부내용
구현언어		Embedded Visual C++3.0
Communication		CDMA 2000 1x
PDA O/S		PPC 2002
PDA	Kind	Poz, iPAQ
	CPU	Xscale 200MHz StrongArm 206MHz
	RAM	64Mbyte
	ROM	64Mbyte
RMS 도메인 서버		Custom Service Server
RMS 서버		Custom Service Server
DB		Custom file type DB

4.2 구현결과

다음은 RMS의 구현 결과로서 그림 14의 좌측은 RMS 에이전트의 메인 화면이고 우측은 리소스 파일을 호출하여 다운로드 받는 과정을 나타낸 화면이다.

그림 14. RMS의 메인 화면과 다운로드 화면

RMS에서 호출하는 리소스 파일의 형태는 네가지로 구분하여 실험하였다. 리소스 파일인 Test.dll과 이미지 파일인 Test.bmp, 그리고 음악 파일인 Test.mp3, 동영상 파일인 Test.wmv 파일을 호출하여 실행하였다.

그림 15는 RMS를 통해서 구현된 결과로서 dll파일, bmp파일을 브라우저를 통해서 나타낸 화면이고 음악 파일은 생략하였다.

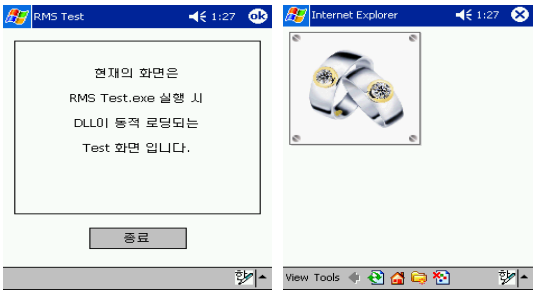

그림15. dll과 bmp 파일을 실행한 화면

그림 16의 좌측은 동영상 파일의 구현 결과이고 우측 그림은 다운로드 받은 파일들이 RMS 스토리지 영역에 존재하고 있는 화면이다. 현재는 RMS 스토리지 영역을 5Mbyte로 설정하였으나 사용자 정의에 의해서 조절이 가능하다.

그림 16. 동영상 실행과 스토리지 영역의 화면

RMS를 적용하기 위한 실험 파일을 무선 접속으로 다운로드받는데 표 4와 같은 시간이 소요되었다.

[표 4] 실험파일 접속 다운로딩 타임

실험파일	사이즈	다운로딩 타임
Test.dll	5KB	0.81 sec
Test.bmp	17KB	2.74 sec
Test.mp3	1.171KB	193.73 sec
Test.wmv	267KB	43.12 sec

접속 로딩 시간은 외부 접속 환경을 고려하여 5회 시도한 평균 시간이다.

메모리 내부에 자원을 가지고 수행하는 기존의 경우보다 필요할 경우 호출하여 수행하는 RMS에서는 CPU 점유율과 파일 수행의 퍼포먼스가 우수한 효과가 있다. RMS를 통한 “모바일 공간 확장”으로 메모리 증대에 대한 제조비용 감소와 기존 사용자들의 무선 인터넷 서비스 확대를 기대할 수 있다.

5. 결론 및 연구계획

무선 인터넷을 제공하는 모바일 디바이스는 이동의 편리성을 가지고 있는 대신에 메모리 용량이 협소하여 효과적인 자원관리를 수행하지 못하는 단점을 가지고 있다. 향후의 기술 개발은 이러한 단점을 개선해 갈 것이다. 그러나 최소화되어야 하는 모바일 디바이스의 특성상 메모리 영역을 무한대로 확장한다는 것은 한계가 있다. 무선 인터넷이 대중화 될 때에는 고객들의 요구 사항은 높아지고 이와 함께 데이터의 용량이 증가되어 현재보다는 개선된 사항들이 또 다시 제약사항으로 변경될 수 있기 때문이다. 따라서 모바일 디바이스의 메모리 공간을 외부의 서버 영역까지 확장하여 사용할 수 있는 모바일 플랫폼 구조로 변경되어야 한다.

본 논문에서는 모바일 디바이스의 미래 지향적인 수행 구조로서 “모바일 공간 확장”을 제안하였다. 모바일 디바이스가 내부 메모리에서 외부의 서버까지 메모리 공간을 확장하여 프로세스 수행이 가능하고 디바이스 내부의 파일을 효율적으로 관리하여 최적의 메모리를 확보할 수 있는 모바일 에이전트로 자

원 관리시스템(RMS: Resource Management System)을 설계하고 구현하였다. 본 논문에서는 RMS를 통해서 에이전트 영역의 설계를 구체화 하였으나 RMS 도메인이나 서버는 일반적인 설계 방법을 적용하여 호출시 발생하는 로드 밸런싱 등을 적용하는 데에는 추가적인 보완이 필요하다.

따라서 향후의 연구 계획은 RMS 에이전트의 기능을 고도화하고 도메인을 포함한 RMS의 서버의 설계 모델을 보다 구체화하며 무선 접속시 트래픽을 최소화 할 수 있는 최적의 방법을 추가로 연구할 계획이다.

참고문헌

[1]James Y. Wilson. Building Powerful Platforms with Windows CE, Aspi Havewala, 2002.

[2]T. Magedanz, "Intelligent Agent", CACM, Vol.37, No.7. pp.18-21, 1994.

[3]Intromobile, Mobile Multimedia Technology Trend,<http://www.intromobile.co.kr/solution/>

[4]Microsoft, PocketPC Official site, <http://www.microsoft.com/mobile/pocketpc/>

[5]Microsoft WinCE, Mobile Solutions, <http://www.microsoft.com/windows/embeded/>

[6]곽용재역, COM/DCOM 플라이머 플러스, 인포북, 1999.

[7]김종완, "PDA용 소프트웨어를 위한 무선 에이전트 클래스의 설계 및 구현", 정보과학회 춘계학술대회 Vol.28. No.1. pp.637-639, 2001.

[8]노영선역, Programming Microsoft WindowsCE 2Edition, 정보문화사, 2001.

[9]안태균외, 3인 포켓 PC와 함께하는 모바일 프로그래밍, 정보게이트, 2002.

[10]이재호, "에이전트 시스템의 연구 및 개발 동향", 정보과학회지, Vol.5. No.5. pp.4-5, 2000.

[11]정보통신진흥국, 무선인터넷 가입자현황 <http://www.mic.go.kr/jsp/mic_d/d700-0002-1.jsp>

나승원

'99.09-'현재 동국대학교 대학원 컴퓨터공학과 프로그램 언어연구실 박사과정

관심분야 : 프로그래밍 언어, 무선인터넷 언어, Java.

오세만

'93.03-'99.02 동국대학교 컴퓨터 공학과 대학원 학과장

'85.03-현재 동국대학교 컴퓨터 공학과 교수

관심분야 : 컴파일러, 프로그래밍 언어, 무선인터넷 언어. Java